2020

Day in the Life of a Data Scientist

EC ENGR 131A FINAL PROJECT

KELLEN CHENG

Discussion 1C
UID: 905-155-544
Kellen Cheng

MATLAB EC ENGR 131A

1. (a) $E_{MMSE} = \mathbb{E}\left[\sum_{i \in K_{miss}} (x_i - a_i)^2\right] = \mathbb{E}\left[\sum_{i \in K_{miss}} (x_i^2 - 2a_i x_i + a_i^2)\right] = \mathbb{E}\left[\sum_{i \in K_{miss}} (x_i^2) - 2\sum_{i \in K_{miss}} a_i x_i + \sum_{i \in K_{miss}} a_i^2\right]$

$= \sum_x \left(\sum_{i \in K_{miss}} x_i^2 - 2\sum_{i \in K_{miss}} a_i x_i + \sum_{i \in K_{miss}} a_i^2\right) P(x)$ $\rightarrow$ Now apply the derivative with respect to a_i:

$= -2\sum_{i \in K_{miss}} x_i + 2na_i = 0$ (setting to 0 to minimize the derivative)

$\Rightarrow a_i = \dfrac{\sum_{i \in K_{miss}} x_i}{n} = \mu$

$\Rightarrow$ Therefore, to minimize E_{MMSE}, $a_i = \mu$ (QED)

(b) Observation: $\hat{\mu}_N$ approaches a value around 20 for large N

(c) For large N, our accuracy gets more and more accurate (therefore it approaches ~100, but never equals it) Exactly: 99.2

(d) Recall: $\lim_{n\to\infty} \hat{\mu}_N = \lim_{n\to\infty} \dfrac{\sum_i x_i}{n} = \mathbb{E}[X] = \mu$

$\Rightarrow$ Therefore, for large N: $\hat{A}_N = \dfrac{\sum_{i \in K_{avail}} (x_i - \hat{\mu}_N)^2}{|K_{avail}|}$

N.B. Let $f(x_i) = (x_i - \hat{\mu}_N)^2$

$\Rightarrow \hat{A}_N = \dfrac{\sum_{i \in K_{avail}} (x_i - \hat{\mu}_N)^2}{|K_{avail}|} = \dfrac{\sum_{i \in K_{avail}} (f(x_i))}{|K_{avail}|} \Rightarrow \lim_{n\to\infty} \hat{A}_N = \mathbb{E}[f(x_i)] = \mathbb{E}[(x_i - \hat{\mu}_N)^2]$

$\Rightarrow$ For large values of N, we see that $\hat{A}_N$ approaches the $VAR(X_i)$

$\Rightarrow$ As variance for a random distribution cannot be 0, we have shown that even for large N, $\hat{A}_N$ is limited to σ^2, which is 99.2 (from the graph)

(e) From part (d), σ^2 is equal to the limiting value of $\hat{A}_N$.

2. (b) For X_i (a uniform continuous random variable from [2, 6])

$\Rightarrow \mu_{X_i} = \frac{1}{2}(2+6) = 4$, $VAR(X_i) = \frac{1}{12}(6-2)^2 = \frac{4}{3}$

$\Rightarrow \mu_{M_n} = \mathbb{E}[M_n] = \mathbb{E}\left[\frac{1}{n}(X_1 + \dots + X_n)\right] = \mu_{X_i} = 4$, $VAR(M_n) = VAR\left(\frac{1}{n}(X_1 + \dots + X_n)\right) = \frac{1}{n}VAR(X_i) = \frac{4}{3n}$

$\Rightarrow$ Therefore: $\mu_{X_i} = \mu_{M_n} = 4$, $VAR(X_i) = \frac{4}{3}$, $VAR(M_n) = \frac{4}{3n}$

N.B. Continued...

Discussion 1C
UID: 905-155-544
Kellen Cheng

MATLAB EC ENGR 131A

2. (d) For X_i (an unfair 5-sided die)

$\Rightarrow \mu_{X_i} = \mathbb{E}[X_i] = (\frac{1}{7})(1+3+5) + (\frac{2}{7})(2+4) = 3$

$\Rightarrow VAR(X_i) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = (\frac{1}{7})(1^2+3^2+5^2) + (\frac{2}{7})(2^2+4^2) - 3^2 = \frac{12}{7}$

$\Rightarrow \mu_{M_n} = \mathbb{E}[M_n] = \mathbb{E}[\frac{1}{n}(X_1+\dots+X_n)]$, $VAR(M_n) = VAR(\frac{1}{n}(X_1+\dots+X_n)) = \frac{1}{n}VAR(X_i) = \frac{1}{3} \times \frac{36}{7n} = \frac{12}{7n}$

$\Rightarrow$ Therefore: $\mu_{X_i} = \mu_{M_n} = 3$, $VAR(X_i) = \frac{12}{7}$, $VAR(M_n) = \frac{12}{7n}$

3. (a) Rewriting via Bayes' Rule: $P(y=0|\vec{x}) = P(\vec{x}|y=0)(\frac{P(y=0)}{P(\vec{x})})$, &

$P(y=1|\vec{x}) = P(\vec{x}|y=1)(\frac{P(y=1)}{P(\vec{x})})$

$\Rightarrow$ Cancelling like terms: $P(\vec{x}|y=0)P(y=0) \geq P(\vec{x}|y=1)P(y=1)$

(3) $\Rightarrow$ Rewritten: $\frac{1}{\sqrt{(2\pi)^2|\Sigma|}} \exp(-\frac{1}{2}(\vec{x}-\mu_0)^T\Sigma^{-1}(\vec{x}-\mu_0))p \geq \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} \exp(-\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1))(1-p)$

(4) $\Rightarrow$ Cancelling like terms: $\exp(-\frac{1}{2}(\vec{x}-\mu_0)^T\Sigma^{-1}(\vec{x}-\mu_0)) \geq \exp(-\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1))$

$\Rightarrow$ Taking a natural log of both sides: $-\frac{1}{2}(\vec{x}-\mu_0)^T\Sigma^{-1}(\vec{x}-\mu_0) \geq -\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1)$

$\Rightarrow$ Expanding: $\vec{x}^T\Sigma^{-1}\vec{x} + \mu_0^T\Sigma^{-1}\mu_0 - 2\mu_0^T\Sigma^{-1}\vec{x} \geq \vec{x}^T\Sigma^{-1}\vec{x} + \mu_1^T\Sigma^{-1}\mu_1 - 2\mu_1^T\Sigma^{-1}\vec{x}$

$\Rightarrow \underbrace{(\Sigma^{-1}(\mu_1-\mu_0))^T}_{\vec{b}}\vec{x} + \underbrace{\frac{1}{2}(\mu_0-\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1)}_{a} \geq 0$ QED

(b) Percentage of samples from class 0: 50.1667 % - 0.3334 % = 49.8333 %

(c) We can proceed from line (4) in part (a), but accounting for the probabilities as shown:

$\exp(-\frac{1}{2}(\vec{x}-\mu_0)^T\Sigma^{-1}(\vec{x}-\mu_0))p_0 \geq \exp(-\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1))p_1$

$\Rightarrow$ Taking a natural log of both sides: $-\frac{1}{2}(\vec{x}-\mu_0)^T\Sigma^{-1}(\vec{x}-\mu_0) + \ln(p_0) \geq -\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma^{-1}(\vec{x}-\mu_1) + \ln(p_1)$

$\Rightarrow$ Expanding & Simplifying: $(\Sigma^{-1}(\mu_1-\mu_0))^T\vec{x} + \frac{1}{2}(\mu_0-\mu_1)^T\Sigma^{-1}(\mu_0-\mu_1) + \ln(\frac{1-p}{p}) \geq 0$

(d) Percentage of samples from class 0: 52 % - 4 % = 48 %

- N.B. Changing p had a notable effect in shifting the count in each class, i.e. the contour line was shifted, redefining points so that some points originally in class 1 were reclassified as belonging to class 0, due to the $\ln(\frac{1-p}{p})$ term.

N.B. Continued...

Discussion 1C
UID: 905-155-544
Kellen Cheng

3. (e) We can proceed from line (3) in part (a), but accounting for differing covariance matrices:

$$\frac{1}{\sqrt{(2\pi)^2|\Sigma_0|}}\exp\left(-\frac{1}{2}(\vec{x}-\mu_0)^T\Sigma_0^{-1}(\vec{x}-\mu_0)\right)p \geq \frac{1}{\sqrt{(2\pi)^2|\Sigma_1|}}\exp\left(-\frac{1}{2}(\vec{x}-\mu_1)^T\Sigma_1^{-1}(\vec{x}-\mu_1)\right)(1-p)$$

$\Rightarrow$ Taking a natural log: $-\frac{1}{2}\ln(|\Sigma_0|) - \frac{1}{2}(\vec{x}-\mu_0)^T\Sigma_0^{-1}(\vec{x}-\mu_0) + \ln(p) \geq$

$$-\frac{1}{2}\ln(|\Sigma_1|) - \frac{1}{2}(\vec{x}-\mu_1)^T\Sigma_1^{-1}(\vec{x}-\mu_1) + \ln(1-p)$$

$\Rightarrow$ Simplifying: $\ln\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right) + \frac{2}{2}(\vec{x}-\mu_1)^T\Sigma_1^{-1}(\vec{x}-\mu_1) - \frac{2}{2}(\vec{x}-\mu_0)^T\Sigma_0^{-1}(\vec{x}-\mu_0) + \ln\left(\frac{p}{1-p}\right)$

$\Rightarrow$ Further Simplifying: $\vec{x}^T(\Sigma_0^{-1} - \Sigma_1^{-1})\vec{x}\left(\frac{1}{2}\right) + (\Sigma_1^{-1}\mu_1 - \Sigma_0^{-1}\mu_0)^T\vec{x}$

$$+\frac{1}{2}\left((\mu_0^T\Sigma_0^{-1}\mu_0 - \mu_1^T\Sigma_1^{-1}\mu_1) + \ln\left(\frac{|\Sigma_0|}{|\Sigma_1|}\right) + 2\ln\left(\frac{1-p}{p}\right)\right) \geq 0$$

(f) Percentage of samples from class 0: 50.4071% - 0.8142% = 49.5929%

Appendix A (MATLAB Figures)

Figure 1 (Problem 1 Part B)

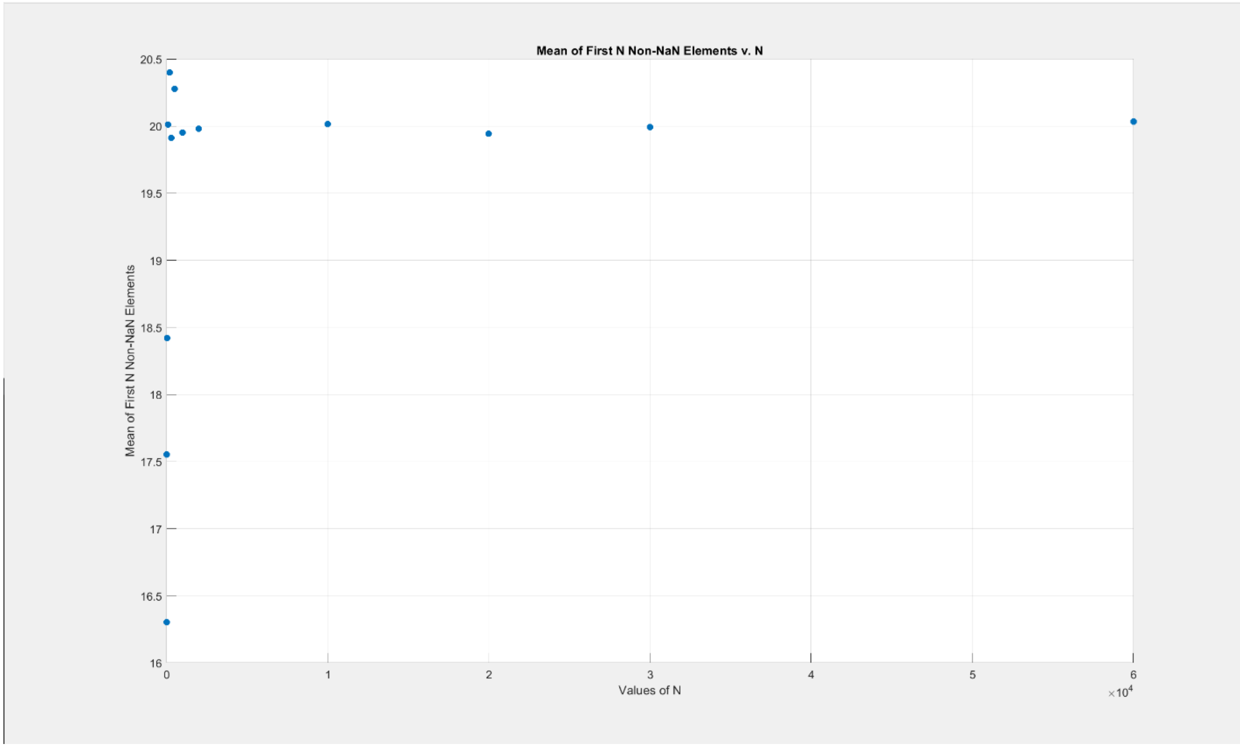


Figure 2 (Problem 1 Part C)

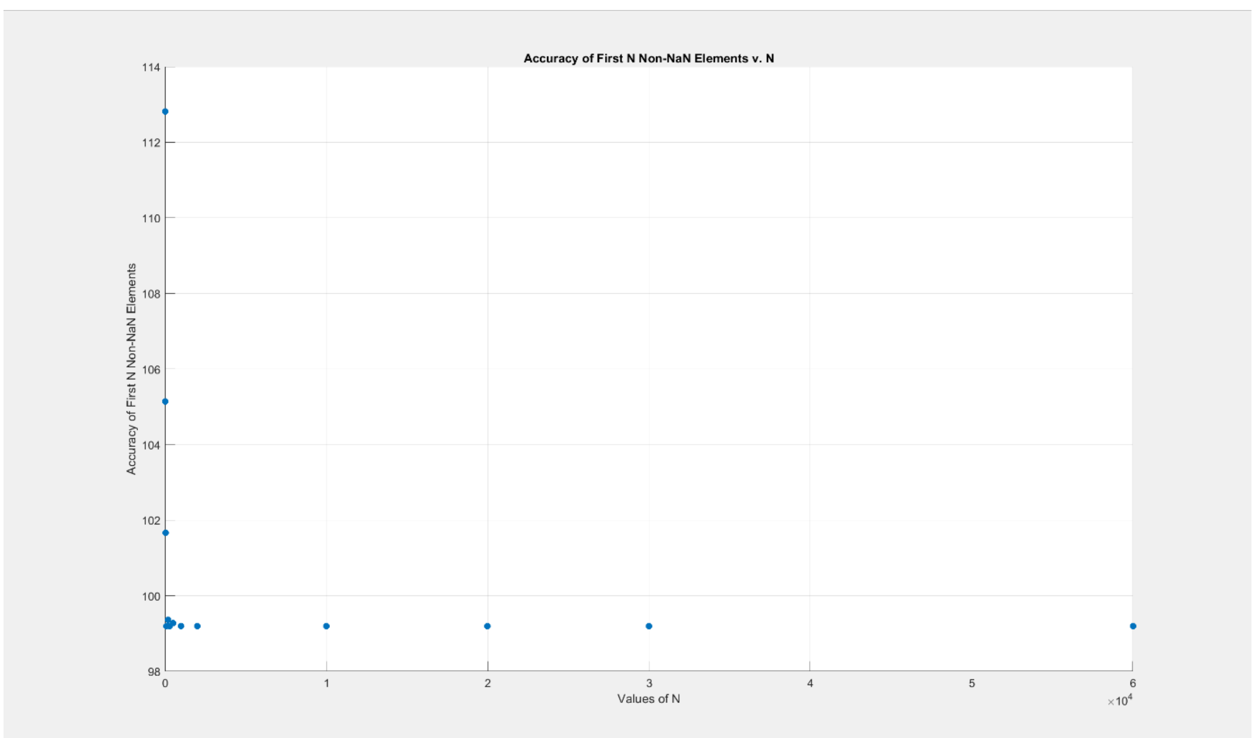


Figure 3 (Problem 2A – PDF)

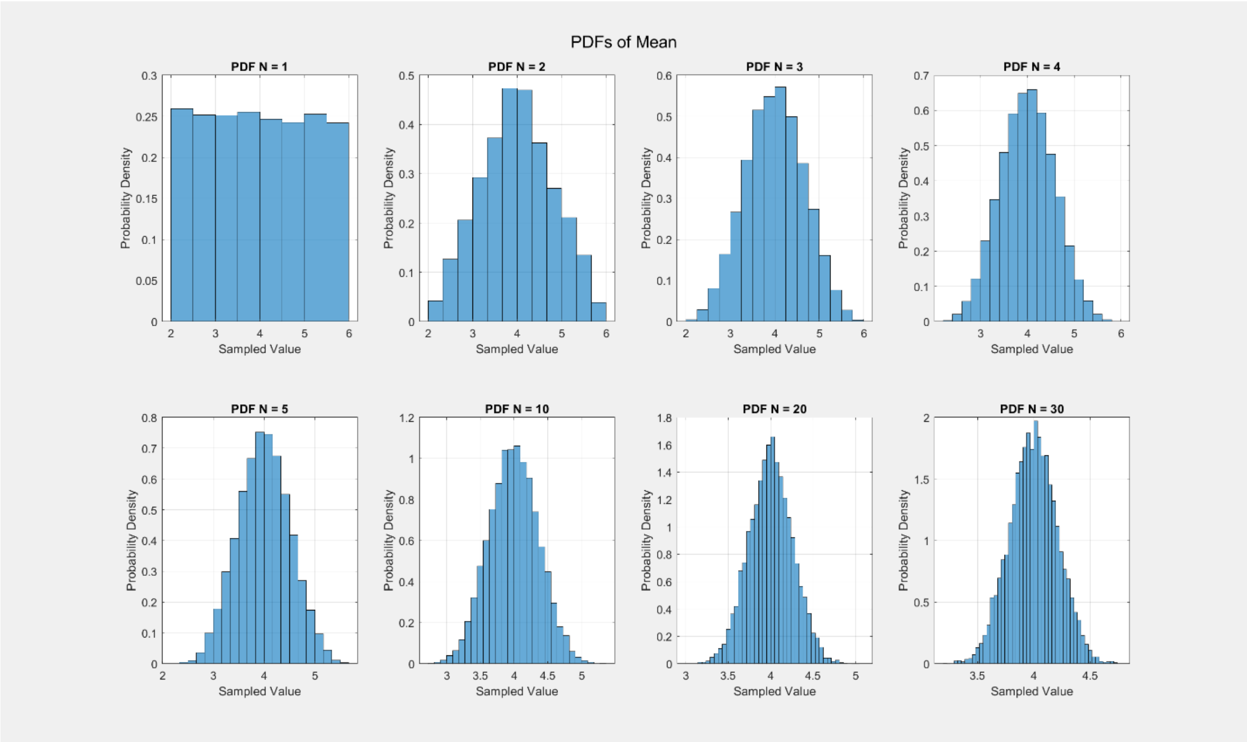


Figure 4 (Problem 2A – CDF)

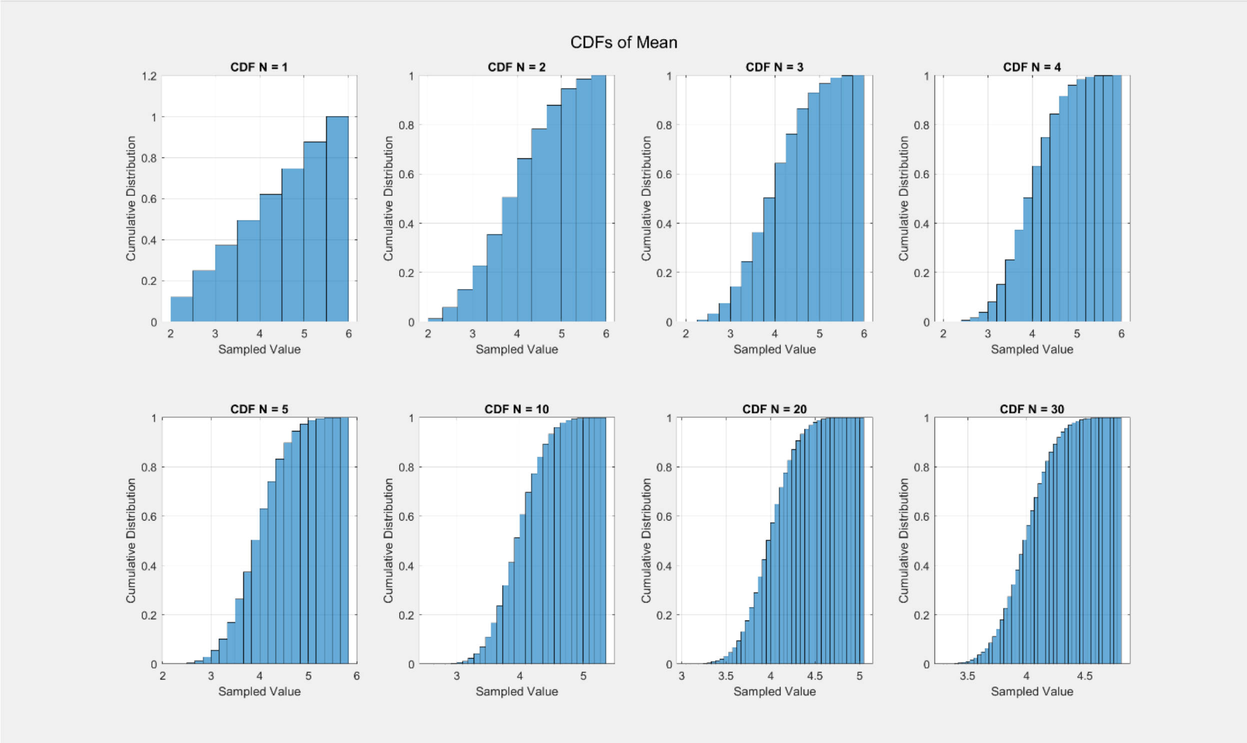


Figure 5 (Problem 2C – PDF)

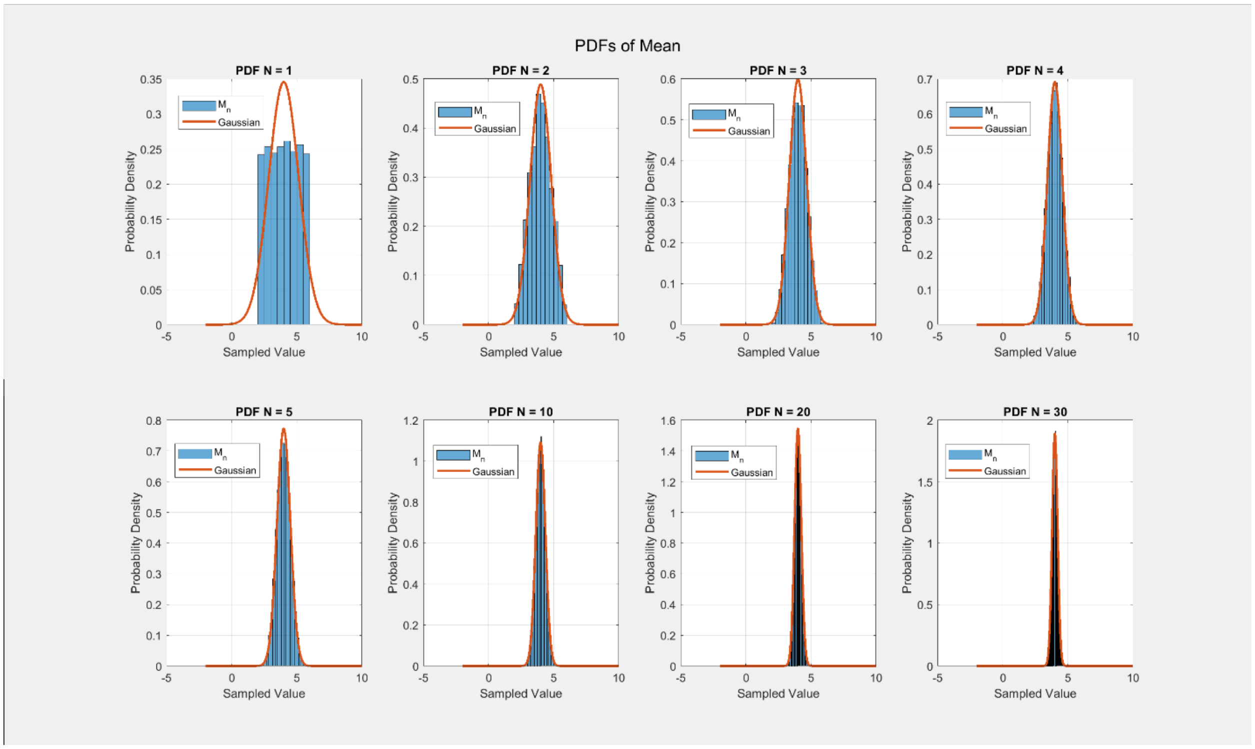


Figure 6 (Problem 2C – CDF)

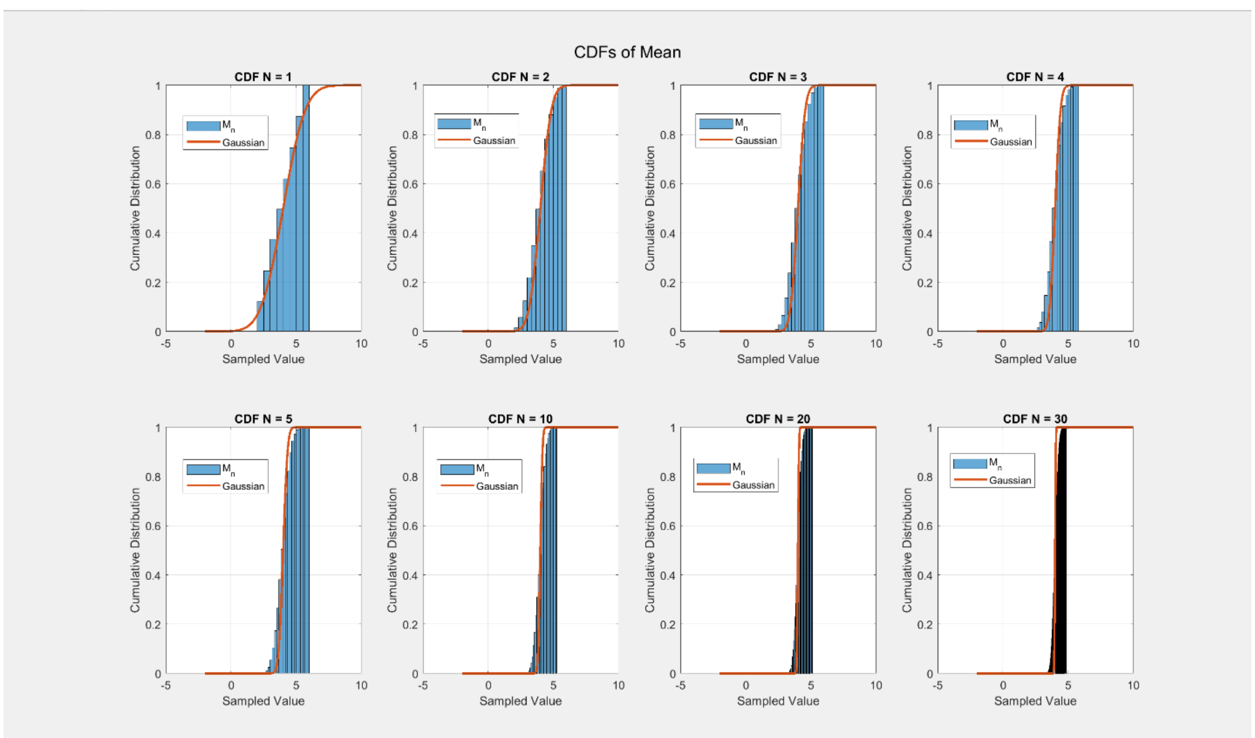


Figure 7 (Problem 2D, A – PDF)

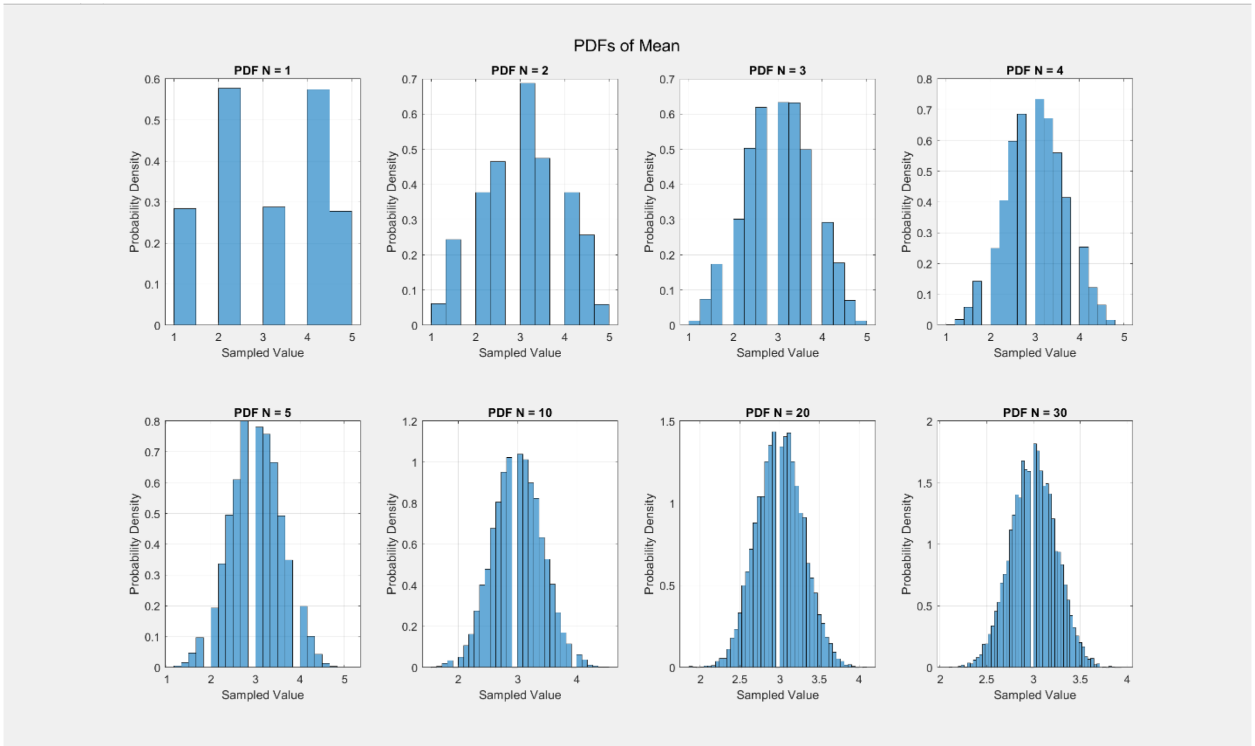


Figure 8 (Problem 2D, A – CDF)

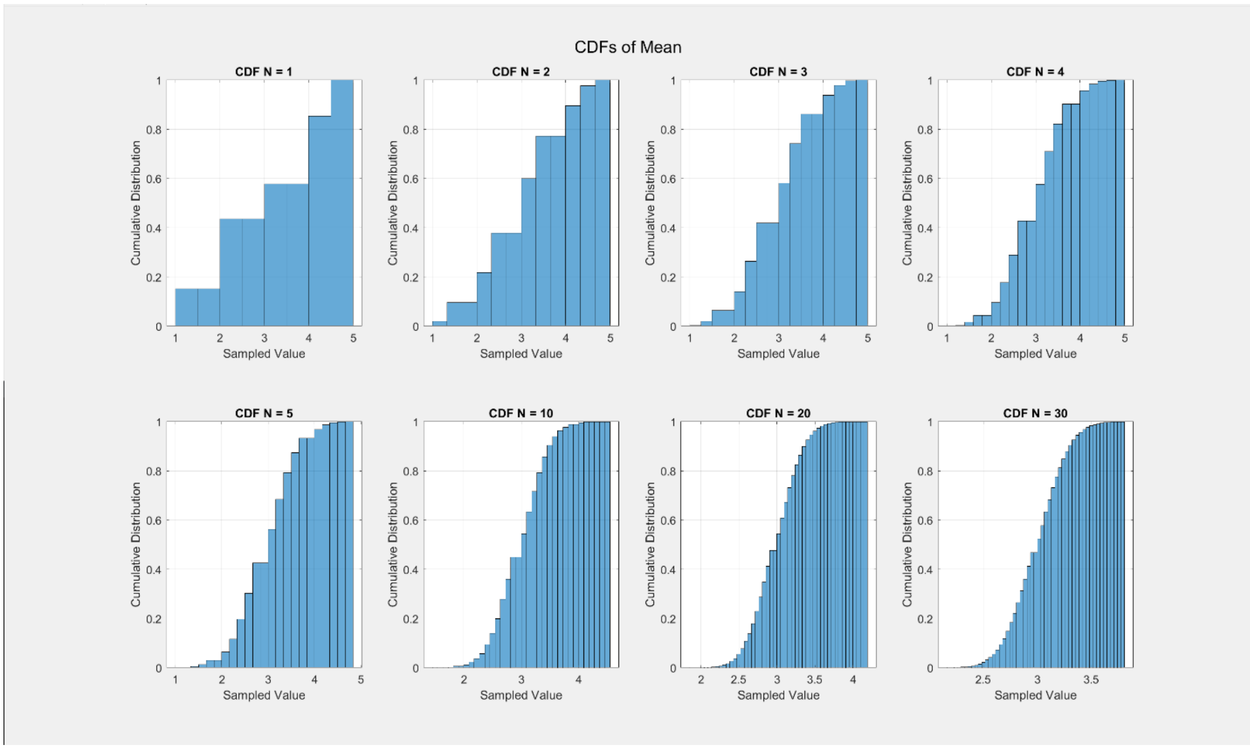


Figure 9 (Problem 2D, C – PDF)

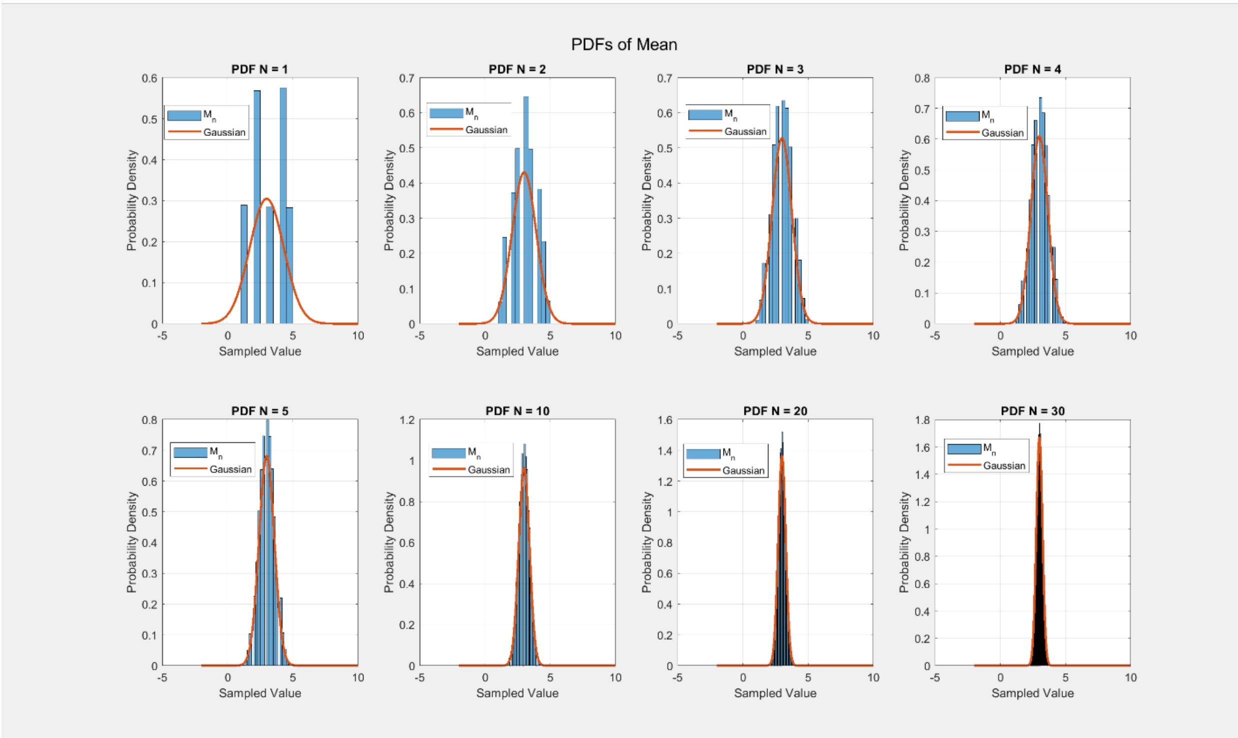


Figure 10 (Problem 2D, C – CDF)

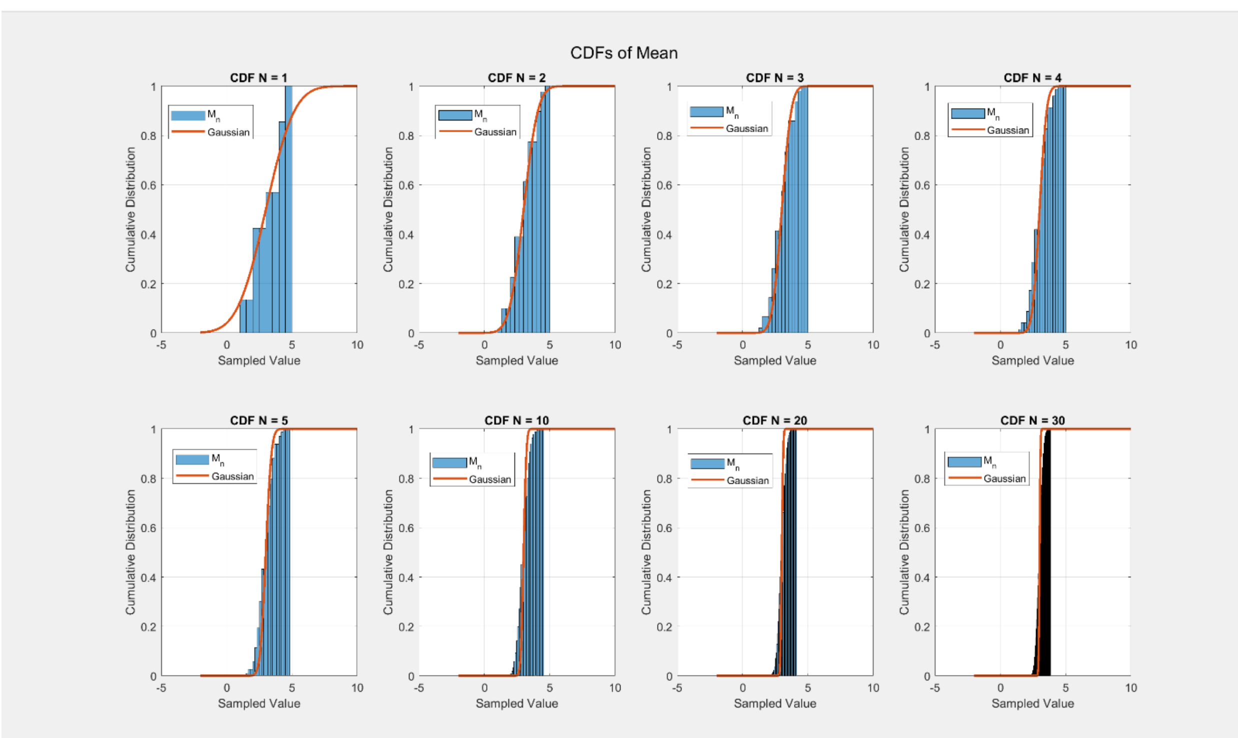


Figure 11 (Problem 3 Part B)

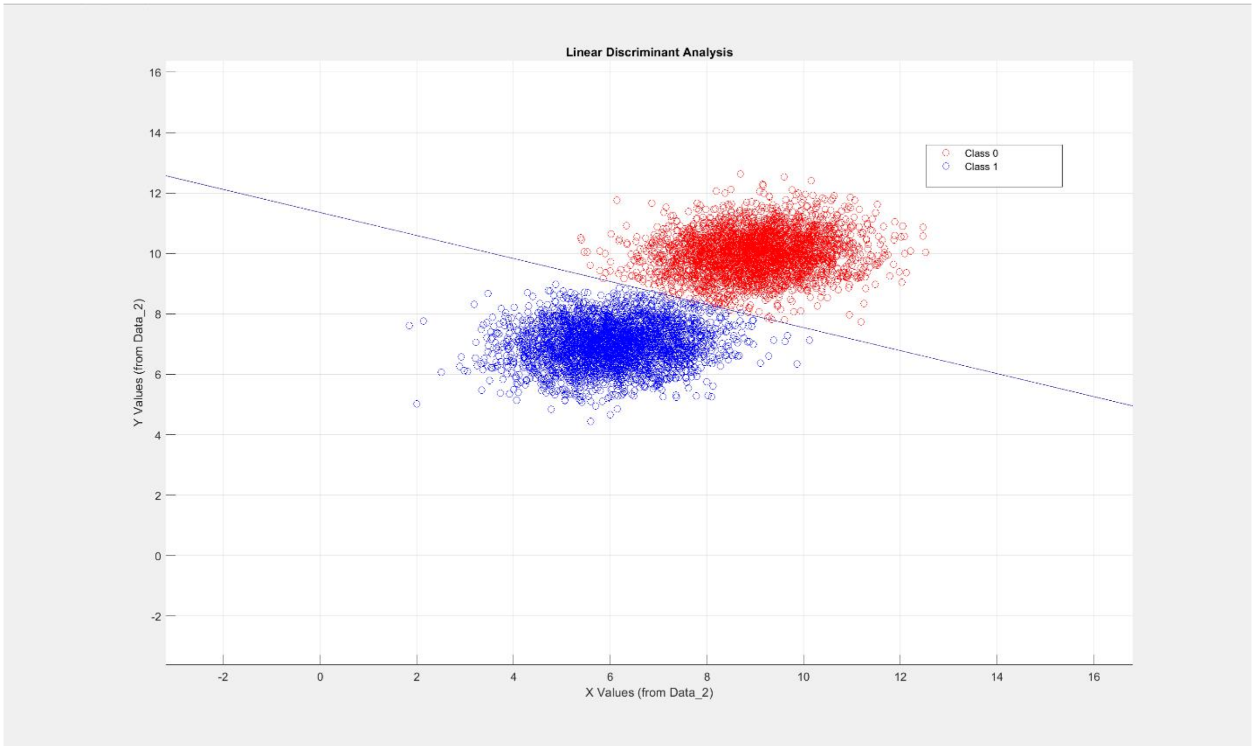


Figure 12 (Problem 3 Part D)

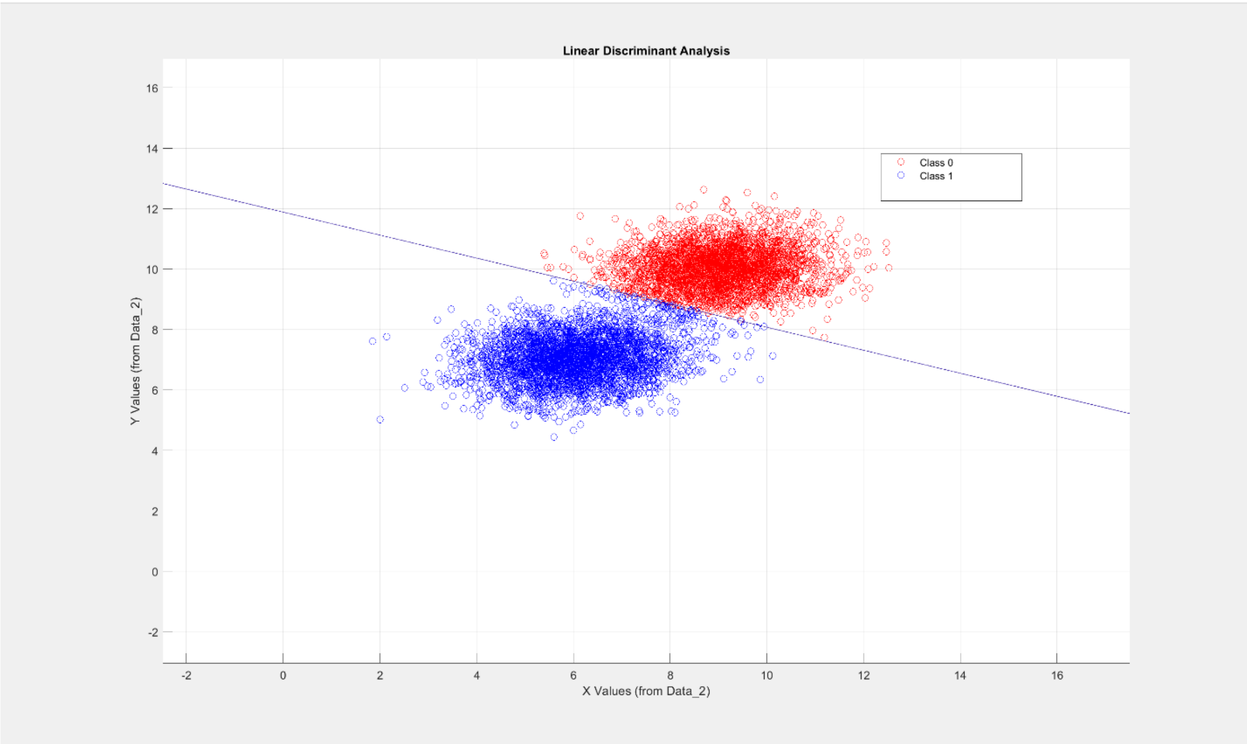
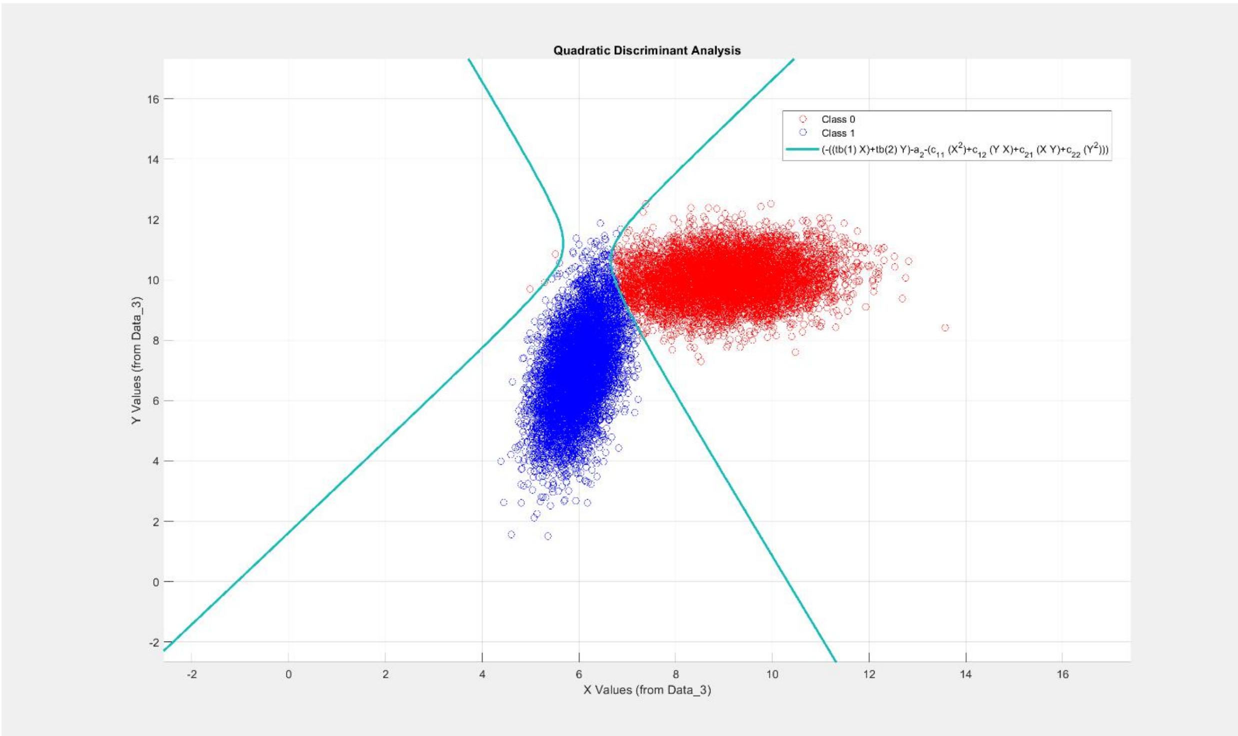


Figure 13 (Problem 3 Part F)



Appendix B (MATLAB Codes)

Code 1 (Problem 1 Part B)

```
%% Problem I Part B

% Declares array with N values and array to hold mu values
countArray = [10 20 50 100 200 300 500 1000 2000 10000 20000 30000 60000];
storageArray = [];

% Filters out NaN values in data.txt
arrayA = importdata('data.txt');
farrayA = arrayA(~isnan(arrayA));

n = 1; % Indexing variable
total = 0; % Keeps track of sum of data points

for i = 1:length(countArray)
    while n < (countArray(i) + 1)
        total = total + farrayA(n); % Adds to total
        n = n + 1; % Increments index
    end

    mu = total / countArray(i); % Calculates mu
    storageArray = [storageArray, mu]; % Appends mu to array
end

% Plots mu_N v. N values
scatter(countArray, storageArray, 'filled'); grid on;
title('Mean of First N Non-NaN Elements v. N'); xlabel('Values of N');
ylabel('Mean of First N Non-NaN Elements');
```

Code 2 (Problem 1 Part C)

```
%% Problem I Part C

% Declares array with N values and array to hold A_N values
countArray = [10 20 50 100 200 300 500 1000 2000 10000 20000 30000 60000];
accArray = [];

% Filters out NaN values in data.txt
arrayA = importdata('data.txt');
farrayA = arrayA(~isnan(arrayA));

for i = 1:length(countArray)
    total = 0; % Resets total on each iteration
    for k = 1:95241 % K_avail = 95241
        % Recall storageArray from Problem I Part B
        total = total + (farrayA(k) - storageArray(i)) .^ 2;
    end

    acc = total / length(farrayA); % Calculates A_N
    accArray = [accArray, acc]; % Appends A_N to array
end

% Plots A_N v. N Values
scatter(countArray, accArray, 'filled'); grid on;
title('Accuracy of First N Non-NaN Elements v. N'); xlabel('Values of N');
ylabel('Accuracy of First N Non-NaN Elements');
```

Code 3 (Problem 2 Part A, Part C)

```
%% Problem II Part A, C

% Declares array with N values and array to hold M_n values
secondN = [1 2 3 4 5 10 20 30];
meanArray = ones(8, 10000);

% Sets parameters for superimposed Gaussian RV (Part C)
x1 = [-2:0.01:10];
mu = 4;

for i = 1:length(secondN)
    % Generates N by 10000 samples of a uniform RV from [2, 6]
    randX = 2 + 4 * rand(secondN(i), 10000);

    % Appends M_n values to array
    meanArray(i, :) = sum(randX, 1) ./ secondN(i);

    subplot(2, 4, i);

    %%% PDF Plots
    histogram(meanArray(i, :), 'BinWidth', (1 / (secondN(i) + 1)), ...
        'normalization', 'pdf');
    titleString = strcat({'PDF N = '}, num2str(secondN(i)));
    title(titleString); xlabel('Sampled Value'); grid on;
    ylabel('Probability Density');

    %%% PDF Superimposed Gaussian
    hold on;
    varianceM = 4 / (3 * secondN(i)); % Sets the variance

    p1 = (1 / (sqrt(2 * pi * varianceM)));
    p2 = exp((-1 / 2) .* (((x1 - mu) .^ 2) / (varianceM)));
    fPdf = p1 .* p2; % PDF of superimposed Gaussian RV

    plot(x1, fPdf, 'LineWidth', 2); hold off; legend('M_{n}', 'Gaussian');

    %%% CDF Plots
    histogram(meanArray(i, :), 'BinWidth', (1 / (secondN(i) + 1)), ...
```

Code 4 (Problem 2 Part A, Part C Cont.)

```
        'normalization', 'cdf');
    titleString = strcat({'CDF N = '}, num2str(secondN(i)));
    title(titleString); xlabel('Sampled Value'); grid on;
    ylabel('Cumulative Distribution');

    %%% CDF Superimposed Gaussian
    hold on;
    fCdf = normcdf(x1, mu, varianceM); % Recall variance from above

    plot(x1, fCdf, 'LineWidth', 2); hold off; legend('M_{n}', 'Gaussian');
end

% PDF Subplot Title
sgtitle('PDFs of Mean');

% CDF Subplot Title
sgtitle('CDFs of Mean');
```

Code 5 (Problem 2 Part D)

```
%% Problem II Part D

% Declares array with N values and array to hold M_n values
secondN = [1 2 3 4 5 10 20 30];
meanArray = ones(8, 10000);
randDieX = [1 2 2 3 4 4 5]; % Weighted array of die values

% Sets parameters for superimposed Gaussian RV (Part C)
x1 = [-2:0.01:10];
mu = 3;

for i = 1:length(secondN)
    % Creates array to hold sampled values
    randXArray = ones(secondN(i), 10000);

    % Generates N by 10000 samples of die values
    for k = 1:secondN(i)
        % Generates 1 by 10000 samples of die values
        randX = datasample(randDieX, 10000);
        randXArray(k, :) = randX; % Appends sampled values to array
    end

    % Appends M_n values to array
    meanArray(i, :) = sum(randXArray, 1) ./ secondN(i);

    subplot(2, 4, i);

    %%% PDF Plots
    histogram(meanArray(i, :), 'BinWidth', (1 / (secondN(i) + 1)), ...
        'normalization', 'pdf');
    titleString = strcat({'PDF N = '}, num2str(secondN(i)));
    title(titleString); xlabel('Sampled Value'); grid on;
    ylabel('Probability Density');

    %%% PDF Superimposed Gaussian
    hold on;
    varianceDie = 12 / (7 * secondN(i)); % Sets the variance
```

Code 6 (Problem 2 Part D Cont.)

```
    p1 = (1 / (sqrt(2 * pi * varianceDie)));
    p2 = exp((-1 / 2) .* (((x1 - mu) .^ 2) / (varianceDie)));
    fPdf = p1 .* p2; % PDF of superimposed Gaussian RV

    plot(x1, fPdf, 'LineWidth', 2); hold off; legend('M_{n}', 'Gaussian');

    %%% CDF Plots
    histogram(meanArray(i, :), 'BinWidth', (1 / (secondN(i) + 1)), ...
        'normalization', 'cdf');
    titleString = strcat({'CDF N = '}, num2str(secondN(i)));
    title(titleString); xlabel('Sampled Value'); grid on;
    ylabel('Cumulative Distribution');

    %%% CDF Superimposed Gaussian
    hold on;
    fCdf = normcdf(x1, mu, varianceDie); % Recall variance from above

    plot(x1, fCdf, 'LineWidth', 2); hold off; legend('M_{n}', 'Gaussian');
end

% PDF Subplot Title
sgtitle('PDFs of Mean');

% CDF Subplot Title
sgtitle('CDFs of Mean');
```

Code 7 (Problem 3 Part B, Part D)

```
%% Problem III Part B, D

% Sets parameters from problem
samples = importdata('data_2.txt'); % Extracts data
mean0 = transpose([9 10]); % Initializes mu_0
mean1 = transpose([6 7]); % Initializes mu_1
tmean0 = transpose(mean0); % Transpose matrix of mean0
tmean1 = transpose(mean1); % Transpose matrix of mean1

% Initializes sigma matrix from problem
sigmaMatrix(1, :) = [1.15 0.1];
sigmaMatrix(2, :) = [0.1 0.5];
sigInverse = inv(sigmaMatrix); % Inverse matrix of sigmaMatrix

% Initializes arrays to hold class data points
class0 = [];
class1 = [];

% Initializes inequality terms
a = (1 / 2) * (tmean0 * sigInverse * mean0 - tmean1 * sigInverse * mean1);
a2 = a + (log(0.95 / 0.05)); % The 'a' term for part (d)

b = (sigInverse) * (mean1 - mean0);
tb = transpose(b); % Transpose matrix of b

% Initializes counters to keep track of class size
counter0 = 1;
counter1 = 1;

for i = 1:length(samples)
    x = samples(i, :); % Takes in one coordinate set

    %%% Part B Linear Inequality
    y = dot(transpose(b), x) + a;

    %%% Part D Linear Inequality
    y = dot(transpose(b), x) + a2;

    if (y < 0) % Classify as class 0 (based off of my signs)
```

Code 8 (Problem 3 Part B, Part D Cont.)

```
        class0(counter0, :) = x; % Adds coordinates to class 0
        counter0 = counter0 + 1; % Increments class 0 counter
    else % Classify as class 1
        class1(counter1, :) = x; % Adds coordinates to class 1
        counter1 = counter1 + 1; % Increments class 1 counter
    end
end

% Percentage of samples from class 0
disp(((counter0 - 1) / length(samples)) * 100);

scatX0 = class0(:, 1); % X-coordinates from class 0
scatY0 = class0(:, 2); % Y-coordinates from class 0
sct0 = scatter(scatX0, scatY0, 'red'); hold on; % Plots scatter of class 0

scatX1 = class1(:, 1); % X-coordinates from class 1
scatY1 = class1(:, 2); % Y-coordinates from class 1
sct1 = scatter(scatX1, scatY1, 'blue'); hold on; % Plots scatter of class 1

legend([sct0, sct1], {'Class 0', 'Class 1'}); % Plot legend

% Initializes xy values for contour equation
X = [-10:0.01:10];
Y = [-10:0.01:10];

%%% Part B Contour Equation
y1 = @(X, Y) -((tb(1) * X) + tb(2) * Y) - a;

%%% Part D Contour Equation
y1 = @(X, Y) -((tb(1) * X) + tb(2) * Y) - a2;

% Plots contour line
fcontour(y1, 'LevelList', 0); title('Linear Discriminant Analysis'); grid on;
xlabel('X Values (from Data\_2)'); ylabel('Y Values (from Data\_2)');
```

Code 9 (Problem 3 Part F)

```
%% Problem III Part F

% Sets parameters from problem
samples = importdata('data_3.txt'); % Extracts data
mean0 = transpose([9 10]); % Initializes mu_0
mean1 = transpose([6 7]); % Initializes mu_1
tmean0 = transpose(mean0); % Transpose matrix of mean0
tmean1 = transpose(mean1); % Transpose matrix of mean1

% Initializes sigma matrices from problem
sigmaMatrix0(1, :) = [1.15 0.1];
sigmaMatrix0(2, :) = [0.1 0.5];

sigmaMatrix1(1, :) = [0.2 0.3];
sigmaMatrix1(2, :) = [0.3 2];

sigInverse0 = inv(sigmaMatrix0); % Inverse matrix of sigmaMatrix0
sigInverse1 = inv(sigmaMatrix1); % Inverse matrix of sigmaMatrix1

% Initializes arrays to hold class data points
class0 = [];
class1 = [];

% Initializes inequality terms
c = (1 / 2) * (inv(sigmaMatrix0) - inv(sigmaMatrix1));
a = (1 / 2) * (tmean0 * sigInverse0 * mean0 - tmean1 * sigInverse1 * mean1);
a2 = a + (1 / 2) * log((det(sigmaMatrix0)) / (det(sigmaMatrix1)));

b = (sigInverse1 * mean1 - sigInverse0 * mean0);
tb = transpose(b); % Transpose matrix of b

% Initializes counters to keep track of class size
counter0 = 1;
counter1 = 1;

for i = 1:length(samples)
    x = samples(i, :); % Takes in one coordinate set
    tx = transpose(x); % Transpose matrix of x
    y = dot(transpose(b), x) + a2; % Latter terms of quadratic inequality
```

Code 10 (Problem 3 Part F Cont.)

```
    % First term of quadratic inequality
    cDot = (c * transpose(x));
    cDot2 = dot(transpose(x), cDot);
    y = y + cDot2;

    if (y < 0) % Classify as class 0 (based off of my signs)
        class0(counter0, :) = x; % Adds coordinates to class 0
        counter0 = counter0 + 1; % Increments class 0 counter
    else
        class1(counter1, :) = x; % Adds coordinates to class 1
        counter1 = counter1 + 1; % Increments class 1 counter
    end
end

% Percentage of samples from class 0
disp(((counter0 - 1) / length(samples)) * 100);

scatX0 = class0(:, 1); % X-coordinates from class 0
scatY0 = class0(:, 2); % Y-coordinates from class 0
sct0 = scatter(scatX0, scatY0, 'red'); hold on; % Plots scatter of class 0

scatX1 = class1(:, 1); % X-coordinates from class 1
scatY1 = class1(:, 2); % Y-coordinates from class 1
sct1 = scatter(scatX1, scatY1, 'blue'); hold on; % Plots scatter of class 1

legend([sct0, sct1], {'Class 0', 'Class 1'}); % Plot legend

% Initializes xy values for contour equation
X = [-10:0.01:10];
Y = [-10:0.01:10];

% Inputs xy values into a single matrix
xyMatrix(:, 1) = X;
xyMatrix(:, 2) = Y;

% Calculates quadratic term of quadratic inequality
a3 = c * transpose(xyMatrix);
a4 = dot(transpose(xyMatrix), a3);
c11 = c(1, 1); c12 = c(1, 2); c21 = c(2, 1); c22 = c(2, 2);
```

Code 11 (Problem 3 Part F Cont.)

```
%%% Part F Contour Equation
y1 = @(X, Y) (-((tb(1) * X) + tb(2) * Y) - a2 - ...
    (c11 * (X .^ 2) + c12 * (Y .* X) + c21 * (X .* Y) + c22 * (Y .^ 2)));

% Plots contour line
fcontour(y1, 'LevelList', 0, 'LineWidth', 2);
xlabel('X Values (from Data\_3)'); ylabel('Y Values (from Data\_3)');
title('Quadratic Discriminant Analysis'); grid on;
```